UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/651,901	08/30/2000	Mariusz H. Jakubowski	MS1-516US	2176

22801 7590 06/17/2005

LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
ARANI, TAGHI T	
ART UNIT	**PAPER NUMBER**
2131	

DATE MAILED: 06/17/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/651,901	JAKUBOWSKI ET AL.	
	Examiner	Art Unit	
	Taghi T. Arani	2131	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *3/21/2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-29,31-34 and 36-44* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-29,31-34 and 36-44* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 August 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-29, 31-34 and 36-44 have been examined and are pending

Continued Examination Under 37 CFR 1.114

2. A request for continued examination under 37 CFR 1.1 14, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.1 14, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.1 14. Applicant's submission filed on 03/21/2005 has been entered.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. Claims 1, 2, 14, 16, 18, 29, 32, 34 and 38 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 1, 16, 29, 34 recite the limitation " instructions that modify a register" in lines 6 of claim 1, lines 2-3 of claim 16, line 8 of claim 29 and in lines 4-5 of claim 34. The term "instructions that modify a register" is indefinite. It is not clear whether a register, a hardware, is modified or content of a register is modified.

Claims 1, 16, 29, 34, 38 recite the limitation "modification made to the register" in line 10 of claim 1, line 6 of claim 16, line 13 of claim 29, line 8 of claim 34 and in lines 6-7 of claim 38.

The term "modification to the register" is indefinite. It is not clear whether modification is made to the register, hardware, or the content of the register.

For purpose of applying art, the term " instructions that modify a register" reads instructions that modify the content of a register and the term "modification to the register" reads modification to the content of the register.

The term "possibly" in claims 2,14,18 and 22 is a relative term which renders the claim indefinite. The term "possibly" is not defined by the claim, the specification does not provide a standard for ascertaining the requisite degree, and one of ordinary skill in the art would not be reasonably apprised of the scope of the invention.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-4, 6-7, 10-20, 22-23 and. 26-29, 31-33 and 41-44 are rejected under 35 U.S.C. 103(a) as being unpatentable over prior art of record, Ackerman (USP 6,256,777) in view of USP 5,852,664 to Iverson et al. (hereinafter "Iverson").

As per claim 1, Ackerman teaches one or more computer readable media **(Fig. 1 and associated text, numeral element 16, col. 1, line 47 through col. 2, line 11)** having stored thereon a program that, when executed by one or more processors, causes the one or more processors to perform acts including:

identifying a plurality **(i.e. records a list)** of key instructions in a function **(i.e. locations in machine code where variables need to be saved, compiler 22, col. 4, lines 41-44)**;

inserting into the function, for each of the plurality of key instructions, an extra instruction that modifies a register **(Fig. 3, numeral element 68, col. 4, lines 56-57, debugger 28 placing hidden breakpoints into the machine code that modifies a register, the action may be a manipulation of registers) (col. 4, lines 47-48))** based at least in part on the corresponding key instruction **(i.e. debugger using debug information file which lists all such breakpoints) (col. 4, lines 61-63))**;

Ackerman does not disclose but Iverson discloses **(Fig. 4 and associated text]** a identifying a set of inputs to the function **(numeral element 402, i.e. encoder receives access word from application (col. 6, lines 16-17) to the current frame stored in the memory (see sol. 4, lines 35-38) to generate encoded data (see also numeral element 112 and associated text))**; and

determining a checksum for the function based at least in part on modifications made to the register by the extra instructions when the function is executed with the set of inputs **(numeral element 410, col. 6, lines 51-56, i.e. a hash function is applied to the access word and the checksum value generated by frame data encoder see col. 17, lines 60-67)**.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the Applicant's invention to modify the system of Ackerman with the teachings of Iverson to include identifying a set of inputs to the function and determining a checksum for the function based at least in part on modifications made to the register by the extra

instructions when the function is executed with the set of inputs with the motivation to provide a mechanism which controls a user's access to files (programs or functions) with different versions associated with different skill levels (col. 1, lines 12-37) and to control the access a user has to decode end/or edit files or functions.

As per claims 12-13, Ackerman teaches a method comprising:

identifying a plurality of **(Fig. 3 and associated text, numeral element 60, i.e. records a list)** key instructions in a function **(i.e. locations in machine code where variables need to be saved) (col. 4, lines 41-44))**;

inserting into the function, for each of the plurality of key instructions, an extra instruction that modifies a register based at least in part on the corresponding key instruction **(i.e. placing hidden breakpoints into the machine code (col. 4, lines 56-57) that modifies a register, i.e. the action might tie a manipulation of registers) (col. 4, lines 47-48) based at least in part on the corresponding key instruction, i.e. debugger using debug information file which lists all such breakpoints (col. 4, lines 61-63)**;

wherein the identifying comprises identifying, as a key instruction, each instruction in the function that possibly modifies a register or a flag **(see col. 4, lines 41-44)**.

Ackerman does not disclose but Iverson discloses generating a checksum on bytes of a digital good based on modifications made by the digital good rather than on reading the bytes **(col. 6, lines 51-67)**, wherein the generating comprises:

identifying a set of inputs to the function **(numeral element 402, i.e. encoder receives access word from application (col. 6, lines 16-17))**; and

determining a checksum for the function by mapping contents of the register to the set of inputs **(Fig. 4 and associated text, numeral element 410, col. 6, lines 51-56, a hash function is applied to the access word and the checksum value generated by frame data encoder see col. 17, lines 60-67)**.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the Applicant's invention to modify the system of Ackerman with the teachings of Iverson to include identifying a set of inputs to the function and determining a checksum for the function by mapping contents of the register to the set of inputs with the motivation to provide a mechanism which controls a user's access to files (programs or functions) with different versions associated with different skill levels [col. 1, lines 12-37] and to control the access a user has to decode end/or edit files or functions.

As per claim 16, Ackerman teaches a method comprising:

inserting into a segment of a digital good, a plurality of instructions that modify a register **(Figure 3, numeral element 64, col. 4, lines 56-57, i.e. placing hidden breakpoints into the machine code)** that modifies a register **(i.e. the action might tie a manipulation of registers) (col. 4, lines 47-48))** based at least in part on the corresponding key instruction **(i.e. debugger using debug information file which lists all such breakpoints) (col. 4, lines 61-63)**;

Ackerman does not disclose but Iverson discloses **(Fig. 4 and associated text)** identifying a set of inputs to the segment **(col. 6, lines 16-17, Fig. 4, numeral element 402, i.e. encoder receives access word from application to the segment (see sol. 4, lines 35-38) to generate encoded data and the current frame stored in the memory is modified (numeral element 112 of Fig. 1))**; and

determining a checksum for the segment based at least in part on modifications made to the register by the plurality of instructions when the plurality of instructions are executed with the set of inputs to the segment **(numeral element 410, col. 6, lines 51-56, i.e. a hash function is applied to an access word and a checksum value is generated by frame data encoder see col. 17, lines 60-67)**.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the Applicant's invention to modify the system of Ackerman with the teachings of Iverson to include identifying a set of inputs to the segment and determining a checksum for the segment based at least in part on modifications made to the register by the plurality of instructions when the plurality of instructions are executed with the set of inputs to the segment with the motivation to provide a mechanism which controls a user's access to files (programs or segments) with different versions associated with different skill levels and to control the access a user has to decode end/or edit files (Iverson, col. 1, lines 12-37).

As per claims 15 and 28, the combination of Ackerman and Iverson discloses one or more computer-readable memories comprising computer-readable instructions that, when executed by a processor, direct a computer system to perform the method as recited in claims 13 and 16 **(Iverson, col. 1, line 47 through col. 2, line 11)**.

As per claim 29, Ackerman discloses a production system, comprising:

a memory to store an original program **(see Figure 1, #16)**; and

a production server **(see Figure 1, #10 and associated text)** equipped with an oblivious checking protection tool that is used to augment the original program for protection purposes **(i.e. insert breakpoints, col. 4, lines 44-45)**, the production server

being configured to identify a plurality of segments in the original program and apply oblivious checking to each of the plurality of segments **(i.e. locations in machine code where variables need to be saved or where the sequential flow does not match that of source code) (col. 4, lines 42-44) by**:

inserting, into the segment **(col. 4, lines 56-57, i.e. placing hidden breakpoints into the machine code), a plurality of instructions that modify a register (col. 4, lines 4-48, i.e. the action may be manipulation of registers)**;

Ackerman does not disclose but Iverson discloses **(Fig. 4 and associated text)** an oblivious checking protection tool used for the application of oblivious checking to each of the plurality of segments by:

identifying a set of inputs to the segment (numeral element 402, i.e. encoder receives access word from application **(col. 6, lines 16-17) for the current frame stored in the memory (see sol. 4, lines 35-38) and modifying the current frame in the memory 112 of Fig. 1)**; and

determining a checksum for the segment based at least in part on modifications made to the register by plurality of instructions when the segment is executed with the set of inputs **(numeral element 410, col. 6, lines 51-56, i.e. a hash function is applied to a the access word and the checksum value generated by frame data encoder see col. 17, lines 60-67)**.

Therefore it would have been obvious to one of ordinary skill in the art at the time of the Applicant's invention to modify the system of Ackerman with the teachings of Iverson to include identifying a set of inputs to the segment and determining a checksum for the segment based at least in part on modifications made to the register by the

plurality of instructions when the segment is executed with the set of inputs with the motivation to provide a mechanism which controls a user's access to files (programs or functions) with different versions associated with different skill levels [col. 1, lines 12-37] and to control the access a user has to decode and/or edit files or functions.

As per claims 17 and 31, Ackerman teaches a method and production system as recited in claims 16 and 29 respectively, wherein the inserting comprises:

identifying a plurality of key instructions in the segment **(col. 4, lines 41-44, i.e. a record list and locations in machine code where variables need to be saved)**; and

inserting into the segment, for each of the plurality of key instructions, an extra instruction that modifies a register based at least in part on the corresponding key instruction **(col. 4, lines 56-57, i.e. placing hidden breakpoints into the machine code that modifies a register, col. 4, lines 47-48, i.e. the action may a manipulation of registers based at least in part on the corresponding key instruction, i.e. debugger using debug information file which lists all such breakpoints, col. 4, lines 61-63)**.

As per claims 2, 14, 18 and 32, Ackerman teaches computer readable media, method and a production system as recited in claims 1, 13, 17 and 31 respectively, wherein the identifying a plurality of key instruction comprises identifying, as a key instruction, each instruction in the function (segment) that possibly modifies a register or a flag **(see col. 4, lines 41-44)**.

As per claims 3 and 19, Ackerman teaches computer readable media and method as recited in claims 1 and 17 respectively, wherein the identifying a plurality of key instructions comprises identifying, as the plurality of key instructions, a plurality of

instructions that each modify one or more registers or one or more flags **(see col. 4, lines 41-44)**.

As per claims 4 and 20, Ackerman teaches computer readable media and method as recited in claims 1 and 16 respectively, wherein the inserting comprises;

inserting each extra instruction in a location within the function so that the extra instruction is executed if the corresponding key instruction is executed **(col. 4, lines 65-66, i.e. a hidden breakpoint instruction is inserted immediately above the current machine code instruction)**.

As per claims 6, 22 and 33, Ackerman teaches computer readable media, method and a production system as recited in claims 1, 16 and 29 respectively, wherein the inserting comprises inserting the extra instruction into the function (segment) without altering the control flow of the function (segment) **(see col. 5, lines 12-25)**.

As per claims 7 and 23, Ackerman teaches computer readable media and method as recited in claims 1 and 16 respectively, wherein the inserting comprises inserting, for at least one of the plurality of key instructions, a plurality of extra instructions **(col. 4, lines 47-48, i.e. checkpoints)** that modify one or more registers **(i.e. the, action might be a manipulation of registers. It can be inferred that if mare than one register is to be updated by a key instruction, more than one breakpoint needs to be inserted to save the original values of the registers)**.

As per claims 10 and 26, Ackerman does not teach computer readable media and method as recited in claims 1 and 16 respectively, wherein the checksum comprises both an initial value xo and a calculated value Cks calculated according to:

Start with $x = xo$

Cks := f(xo) XOR xo

For i= 1 to K do x; := g(f(x;-1)) Cks += f(xi) XOR x;

End for

Wherein K is the number of inputs in the set of inputs and g represents the mapping function.

From the above it is understood that the output of one iteration is used to compute the input of the next iteration and that the final value of Cks is a cumulative value of all the previous inputs.

Iverson teaches that the output of one iteration is used to compute the input of the next iteration and that the final value of Cks is a cumulative value of all the previous inputs (col. 6, lines 57-67).

Therefore it would have been obvious to apply the teachings of Iverson to the system of Ackerson to include that the output of one iteration is used to compute the input of the next iteration and that the final value of Cks is a cumulative value of all the previous inputs with the motivation to determine whether the data has been corrupted (see Iverson, col. 3, lines 17-30).

As per claims 11 and 27, Ackerman teaches that the function (digital good comprises) is part of a software program **(see col. 3, lines 16-17, i.e. source code or machine code)**.

As per claims 41 and 42, the combination of Ackerman and Iverson as combined in the rejection of claims 1 and 16, teach the determining the checksum so that if the function is changed the checksum will also change **(Iverson, col. 8, lines 42-46).**

It would have been obvious that the checksum will change because checksum are collision resistant (Iverson, col. 3, lines 25-29).

5. **As per claims 5 and 21** are rejected under 35 U.S.C. 103(a) as being unpatentable over Ackerman in view of Iverson as applied to claims 1 and 16 above, in further view of prior at of record, USP 5,809,306 to Suzuki et al. (hereinafter "Suzuki").

The combination of Ackerman and Iverson teaches the system of claims 1 and 16 as discussed above.

The combination of Ackerman and Iverson does not disclose but Suzuki discloses inserting an extra instruction in a location within the functions so that the extra instruction is executed after a corresponding key instruction is executed **(i.e. a compensation instruction is inserted immediately after a machine language instruction for an arithmetic operation that will possibly cause an overflow) (see col. 15, lines 27-30)).**

Therefore, it would have been obvious to one of ordinary skill in the art to modify the combination of Ackerman and Iverson to include that the inserting comprises inserting each extra instruction in a location within the functions so that the extra instruction is executed after the corresponding key instruction is executed with the motivation to minimize overhead during execution [see Suzuki, col. 9, lines 5-7].

6. **Claims 8-9, 24-25 and 43-44** are rejected under 35 U.S.C. 103(a) as being unpatentable over Ackerman in view of Iverson as applied to claims 1 and 16 above, in further view of prior art of record, USP 6,085,029 to Kolawa et al. (" Kolawa" hereinafter).

The combination of Ackerman and Iverson teaches the system of claims 1 and 16 as discussed above.

As per claims 8 and 24, the combination of Ackerman and Iverson does not disclose but Kolawa discloses a system that relates to automatic instrumentation of software programs **(see col. 1, lines 30-31)**. Kolawa teaches that identifying a set of input comprises identifying a set of input patterns that result in different valid computation paths in the function being taken **(i.e. instrumentation routines can be inserted to automatically generate program inputs to achieve full testing of all flow paths in the executable program) (see col. 17, lines 51-54)**.

Therefore it would have been obvious to one of ordinary skill in the art at the time of the Applicant's invention to modify the combination of Ackerman and Iverson with the teaching of Kolawa to include that identifying a set of input comprises identifying a set of input patterns that result in different valid computation paths in the function being taken with the motivation to achieve full testing of all paths in the executable program (see Kolawa, col. 17, lines 53-54).

As per claims 9 and 25, the combination of Ackerman and Iverson does not teach that the identifying a set of input comprises identifying a set of input patterns to the function that result in all valid computation paths in the function being taken.

Kolawa discloses a system that relates to automatic instrumentation of software programs (see cot. 1, lines 30-31). Kolawa teaches that identifying a set of input comprises identifying a set of input patterns to the function that result in all valid computation paths in the function being taken (i.e. instrumentation routines can be

inserted to automatically generate program inputs to achieve full testing of all flow paths in the executable program) (see cot. 17, lines 51-54).

Therefore it would have been obvious to one of ordinary skill in the art at the time of the Applicant's invention to modify the combination of Ackerman and Iverson with the teaching of Kolawa to include that identifying a set of input comprises identifying a set of input patterns to the function that result in all valid computation paths in the function being taken with the motivation to achieve full testing of all paths in the executable program **(see Kolawa, cot. 17, lines 53-54)**.

As per claims 43, Ackerman and Iverson do not disclose but Kolawa discloses repeating execution of the function with a different input of the set of inputs until the function has been executed with each input of the set of inputs **(i.e. instrumentation routines can be inserted to automatically generate program inputs to achieve full testing of all flow paths in the executable program) (see col. 17, lines 51-54))**.

Therefore it would have been obvious to one of ordinary skill in the art at the time of the Applicant's invention to modify the combination of Ackerman and Iverson with the teaching of Kolawa to repeating execution of the function with a different input of the set of inputs with the motivation to achieve full testing of all paths in the executable program **(see Kolawa, col. 17, lines 53-54)**.

As per claim 44, Ackerman and Iverson do not disclose but Kolawa discloses executing the segment a plurality of times **(col. 17, lines 56-67)**, each execution using a different input of the set of inputs **(i.e. a heuristic method wherein instrumentation used to automatically generate test cases for dynamically testing the program)**.

The examiner supplies the same rationale as provided in claim 43 to combine the combination of Ackerman and Iverson with the teaching of Kolawa.

7. Claims 34, 36-40 are rejected under 35 U.S.C. 103(a) as being unpatentable over York-Smith (USP 5,548,648, issued Aug. 1996) and further in view of USP 5,852,664 to Iverson et al. (hereinafter "Iverson").

As per claims 34,York-Smith teaches client-server system **(col. 1, lines 1-9)**, comprising:

a production server to apply oblivious checking to a program to produce a protected program by **(col. 1, lines 48-60)**:

inserting, into a segment of the program, a plurality of instructions that modify a register **(col. 3, lines 25-42, i.e. control block inserted into the encrypted data block)**;

York-Smith discloses identifying a set of inputs to the segment **(Figure 4 and associated text, numeral elements 400,410, 420, 430, 440 and 40 disclose generating random numbers used as input to identify a segment)**; and

a client to store and execute the protected program, the client being configured to evaluate the protected program to determine whether the protected program has been tampered with **(col. 1, lines 1-10,col.), line 23 through col. 6, line 15, i.e. in a local area network an intended recipient receives and evaluates the protected program using a decryption function which is the inverse of corresponding encryption function)**.

York-Smith does not disclose but Iverson disclose determining a checksum value for the segment based at least in part on modifications made to the register by the plurality of instructions when the segment is executed with the set of inputs **(numeral**

element 410, col. 6, lines 51-56, i.e. a hash function is applied to a the access word and the checksum value generated by frame data encoder see col. 17, lines 60-67).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the Applicant's invention to modify the system of York-Smith with the teachings of Iverson to include identifying a set of inputs to the function and determining a checksum for the function based at least in part on modifications made to the register by the extra instructions when the function is executed with the set of inputs with the motivation to provide a mechanism which controls a user's access to files (programs or functions) with different versions associated with different skill levels and to control the access a user has to decode end/or edit protected program or files[Iverson, col. 1, lines 12-37].

As per claim 36,York-Smith does not disclose but Iverson disclose the client client **(Fig. 2 and associated text show a computer system for decoding signals, see also col. 1, lines 21-34)** is to evaluate the protected program by:

generating a checksum value for each of a plurality of the segments of the program **(Iverson, col. 7, lines 23—29, decoder 50 for processes each frame in the sequence by using an access word (input) for each frame in the sequence until the end of the sequence is reached)** based at least in part on both a set of inputs to the segment and the content of a register **(frame content)** that results from applying the set of inputs to the segments **(Iverson, see Figs. 5 and 6 and associated texts, col. 7, lines 30- 57, i.e. decoder 500 performs decoding each frame by applying a hash function to the access word and the checksum value (modified frame content)**;

comparing, for each of the plurality of segments, the generated checksum

value to a stored checksum value corresponding to the segment **(comparator 510, lines 59-64, compares the result of performing the hash function to the lock word retrieved from the frame header by parser 506)**; and

determining the protected program has been tampered with if the generated checksum value for any one or more of the plurality of segments does not match the stored checksum value for the segment **(col. 7, lines 63-66)**;

The Examiner asserts that comparing a generated checksum value to a stored checksum value to determine whether a program was tampered with is old and well known in the art.

Therefore, It would have been obvious to one of ordinary skill in the art to modify the system of York-Smith to include comparing a generated checksum value to a stored checksum value to determine whether a program was tampered with the motivation to verify integrity of the software program.

As per claims 37, Combination of York-Smith and Iverson teaches a client-server as recited in claim 34, wherein the program comprises a software program **(Iverson, col. 8, lines 26-27, i.e. the copyrights of software package)**.

Claim 38 is computer readable media that substantially implement the system of claims 36, therefore the same rejection applies.

As per claim 39, York-Smith and Iverson teach that the plurality of instruction

further cause the processor (s) to repeat the generating, comparing, and determining for a plurality of segments of the digital good **(Iverson, col. 7, lines 25-30, i.e. the decoder processes each frame until the end of sequence is reached, i.e. generating, comparing and determining for a plurality of segments of the digital**

goods addressed in col. 7 lines 31, 67 of Iverson is for every frame and is repeated until the end of sequence is reached).

It would have been obvious to repeat the generating, comparing and determining of plurality of segments of the digital goods for an overall integrity check.

As per claim 40, Combination of York-Smith and Iverson teach one or more computer readable media as recited in claim 38, wherein the digital good comprises a software program **(Iverson, col. 8, lines 25-27)**.

Conclusion

8. Prior arts made of record, not relied upon:

USP 5,745, 569 to Moskowitz et al. discusses a method for protecting computer code copyrights by encoding the code into a data resource with a digital watermark.

USP 6,782,478 to Probert discloses techniques in coding software by an encoder which encodes a machine executable program or portions of a machine executable program.

USP 6,801,999 to Venkatesan et al. discloses a technique for imparting substantial break-once-run-everywhere (BORE) resistance to passive and active software objects, and for controlling access and use of resulting protected objects by a client computer .

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Taghi T. Arani whose telephone number is (571) 272-3787. The examiner can normally be reached on 8:00-5:30 Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ayaz Sheikh can be reached on (571) 272-3795. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Taghi T. Arani, Ph.D.
Examiner
Art Unit 2131
June 10, 2005